 

**COLLEGE CODE : 9222**

**COLLEGE NAME : THENI KAMMAVAR SANGAM COLLEGE OF**

**TECHNOLOGY**

**DEPARTMENT : B.TECH(IT)**

**STUDENT NM-ID : ADDF1D89B74EA569B304A176DCE92850**

**ROLL NO : 922223205004**

**DATE : 24/09/2025**

**Completed The Project Named As**

**PHASE 2**

**NAME : IBM-NJ-STUDENT GRADING SYSTEM**

**SUBMITTED BY,**

**NAME : ASWATHAMAN M**

**MOBILE NO : 8667799800**

# Solution Design & Architecture

**Title:Student Greading System**

## Tech Stack Selection

## A good tech stack ensures your application is scalable, maintainable, and efficient. For the Student Grading System, we can use the following tech stack:

## Frontend (User Interface):

## React.js – A powerful JavaScript library used to build dynamic and responsive UIs.

## HTML5 & CSS3 – Standard web technologies for structure and styling.

## Bootstrap/Tailwind CSS – For responsive design and pre-styled components.

## Backend (Server Side):

## Node.js – A JavaScript runtime environment used for building scalable server-side applications.

## Express.js – A minimal and flexible Node.js web application framework that provides robust features for API development.

## Database:

## MongoDB – A NoSQL database that stores data in flexible, JSON-like documents.

## OR

## MySQL – A widely used relational database if structured data and SQL queries are preferred.

## Authentication:

## JWT (JSON Web Tokens) – For securely transmitting information between parties and handling user authentication.

## Other Tools:

## Postman – For API testing.

## Git & GitHub – Version control and code collaboration.

## VS Code – Code editor.

## UI Structure / API Schema Design

## UI Structure:

## The user interface should be simple and intuitive. The system may include the following pages/screens:

## Login Page – For teachers/admins to securely access the system.

## Dashboard – Overview of total students, grades submitted, pending evaluations, etc.

## Student Management Page – Add/edit/delete student records.

## Grading Page – Enter, update, or view grades for specific subjects or exams.

## Reports Page – View or download grade reports.

## Settings/Profile Page – Manage user profile and preferences.

## API Schema Design (RESTful APIs):

## Endpoints:

## POST /api/auth/login – Login user

## POST /api/students – Add a new student

## GET /api/students – Get list of students

## GET /api/students/:id – Get student details

## PUT /api/students/:id – Update student info

## DELETE /api/students/:id – Delete student record

## POST /api/grades – Add grades

## GET /api/grades/:studentId – Get grades for a student

## GET /api/reports/:studentId – Generate report

## Data Handling Approach

## Efficient data handling is crucial for system reliability, security, and speed.

## Frontend to Backend:

## Data is collected through forms (e.g., student details, marks).

## Input validation is performed on the client side (e.g., required fields, data formats).

## Data is sent to the backend using HTTP requests via Axios or Fetch API.

## Backend Processing:

## Data is validated again (server-side validation).

## If valid, data is saved to the database using database drivers or ORM (like Mongoose for MongoDB).

## Error handling is done using middleware in Express.js.

## Database Layer:

## Student records, grades, and user credentials are stored in collections/tables.

## Relationships (if using SQL) or references (if using NoSQL) are created between students and grades.

## Security:

## Passwords are hashed (e.g., using bcrypt).

## Authentication tokens (JWT) are used to authorize API access.

## Input sanitization is applied to prevent SQL Injection/XSS.

## Data Flow Example:

## Teacher logs in.

## Token is stored in local storage/session.

## Teacher navigates to add grades.

## Submits grade form.

## Backend validates and stores data.

## Response is sent back (success or error).

## Component / Module Diagram

## The system can be divided into the following modules:

## 1. Authentication Module:

## Handles login, token generation, and access control.

## 2. Student Management Module:

## Manages student data (add/edit/delete/search).

## 3. Grade Management Module:

## Adds, updates, and retrieves grade records.

## 4. Report Generation Module:

## Compiles grades and generates summaries or printable reports.

## 5. Admin Dashboard Module:

## Provides analytics and overview of system data.

## Basic Flow Diagram

## This diagram describes the basic flow of how a user (teacher/admin) interacts with the system.

## Flow Description:

## User Login

## Enters credentials.

## System verifies and returns token.

## Dashboard Access

## Token allows access to protected routes.

## Student Management

## Add/Edit/Delete student data.

## Grade Entry

## Enter marks for each student by subject.

## Data is validated and saved.

## Report Generation

## System calculates averages/grades.

## Report can be viewed/downloaded.